ECE 332 – Embedded Systems Lab

Lab 4: HW/SW Compression and Decompression of Captured Image

Objectives

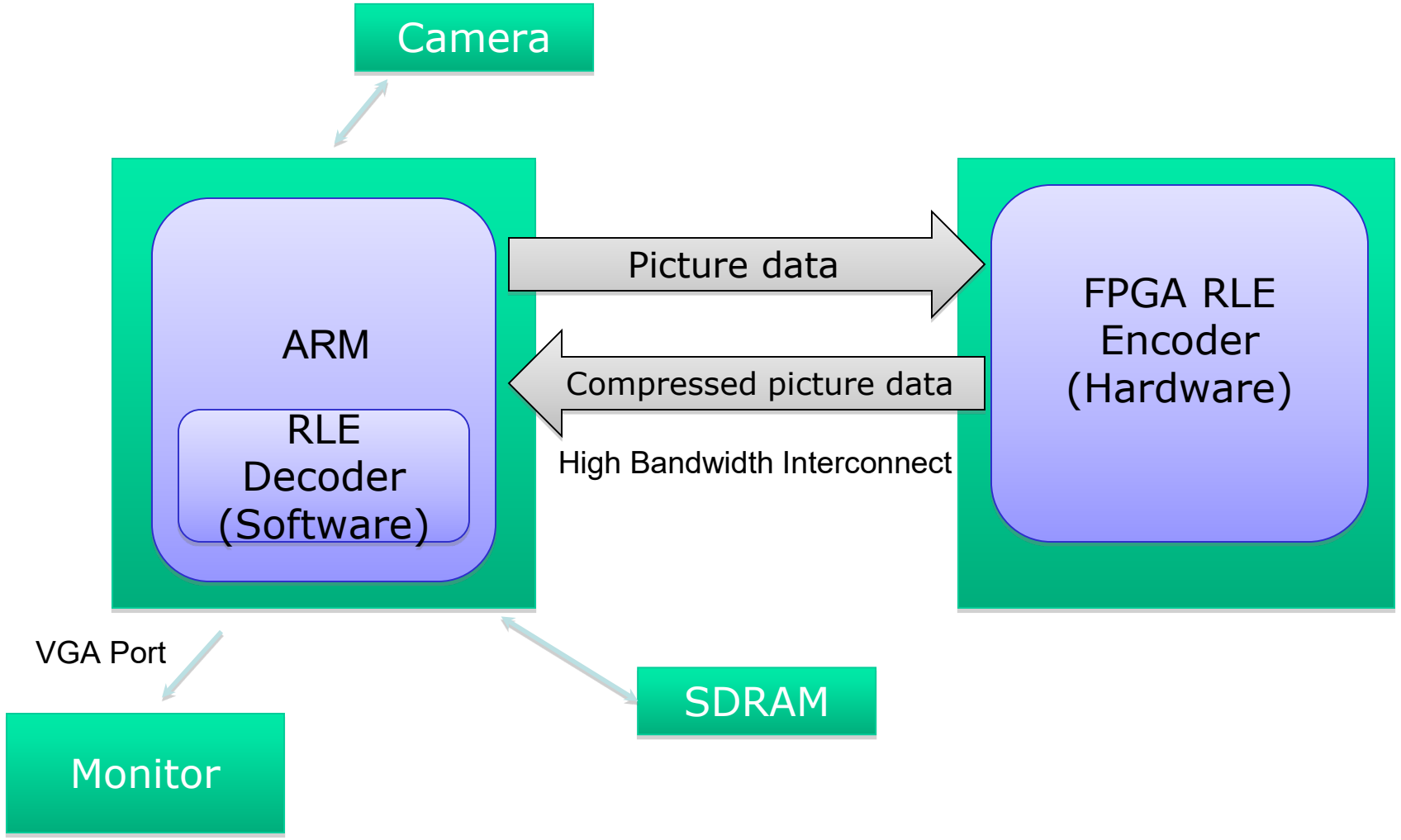
- Understand how to leverage the FPGA-based hardware acceleration in an SoC design.
- Understand how communication is done between the FPGA fabric and the ARM hard core processor system (HPS) in an SoC board
- Understand how to add new components to an existing QSYS design to implement new functionalities

Overview

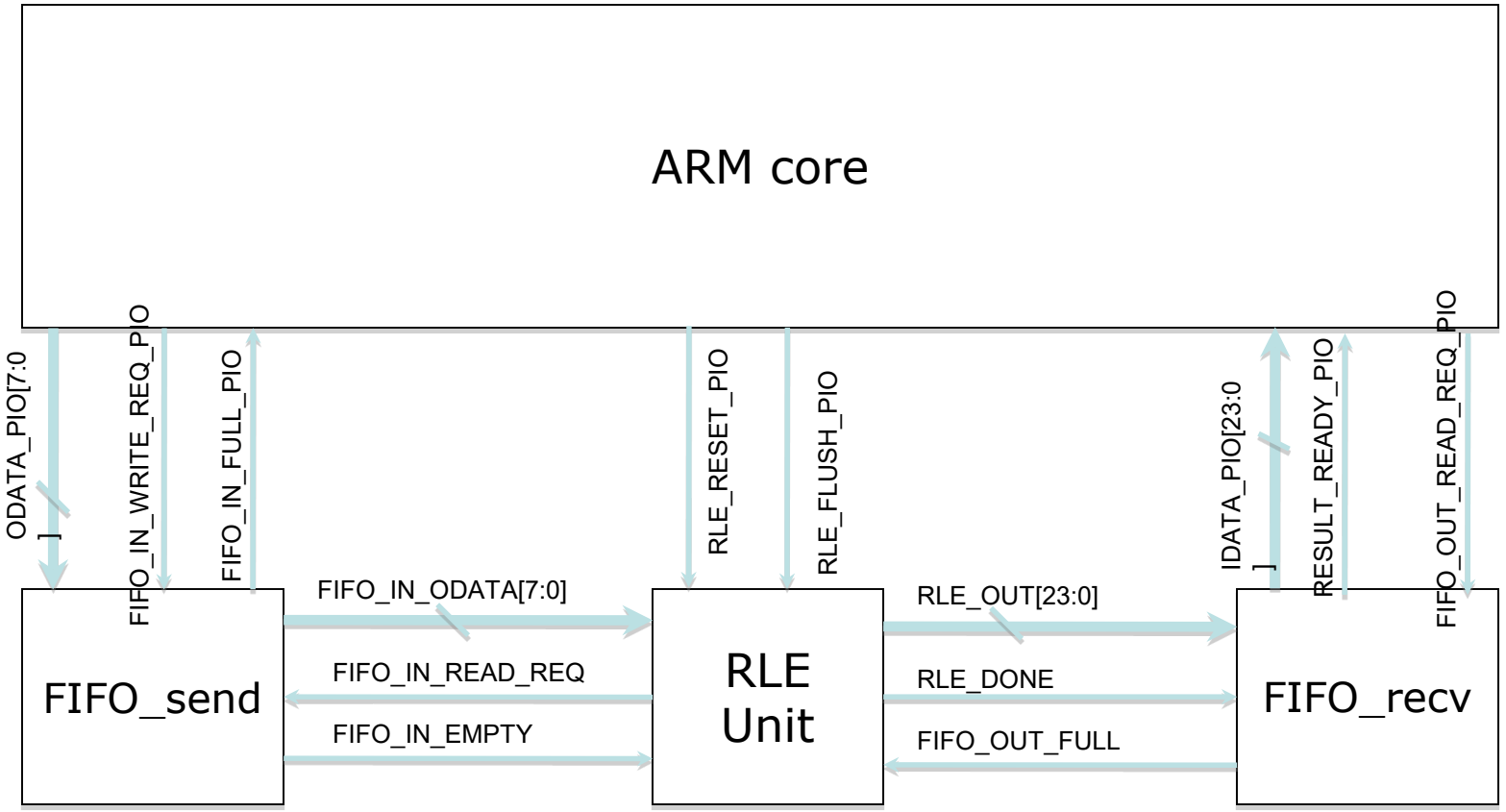
In this lab, you will do the following:

- Implement new features by adding Verilog files into an existing Quartus project
  - FIFO Buffer Modules
  - RLE Compression Module (implemented in lab 3)
- Use the RLE implemented on FPGA fabric to compress the captured image
- Implement the RLE decompression algorithm using C on ARM HPS to recreate the original image and display on the monitor

System Diagram



RLE Hardware Block Structure and Integration



FIFO Buffer

- Saves data based on clock tick when write request signal is asserted
- Holds buffered values in registers until requested from RLE
- Outputs data in a specified way
  - Same order in which it came into the buffer
  - First in the buffer, first to come out (FIFO)
  - Sends data only when requested by a signal
- Used due to asynchronous signals and data sent from the software

HPS-FPGA Lightweight Bridge

- The ARM HPS does not have direct access to hardware instantiated on the FPGA fabric
- Communication is done through the high bandwidth memory-mapped bus bridges
- Tie the Avalon memory mapped slave connections of the PIOs to the Lightweight Bridge AXI Master connection in QSYS (detailed procedure provided in the lab instructions manual)

Getting started - Hardware design

- Copy the contents of lab 2 into a new folder and rename it as lab 4
- Add the rle_enc.v that you designed in lab 3 and rle_fifo_8_24.v given to you into your Quartus project
- Add the additional PIOs to the existing QSYS system
- Modify DE1_SOC_With_D5M.v
  - Instantiate your RLE unit and two FIFO modules
  - Add the necessary wires for ARM-module and module-module connections
  - Connect your Verilog hardware and QSYS system

Signals in Block Structure (To be added in QSYS)

- RESULT_READY_PIO
  - Indicates that there is an encoded data segment in the FIFO. Note that this signal is active low since it is tied to the FIFO empty output.
- RLE_FLUSH_PIO
  - Used at the end of the bit-stream. RLE produces the final encoded data segment immediately with the last counting bit value.
- RLE_RESET
  - Signal for initializing RLE encoder. Assert and de-assert at the beginning of the program.
- IDATA_PIO[23:0]
  - Input ports to receive the encoded data.
  - 1 bit for bit ID, 23 bits for representing number of bits.
- ODATA_PIO[7:0]
  - Output ports to send original bit-stream. Data is sent in 8-bit segments.
- FIFO_IN_FULL_PIO
  - Indicates FIFO is full. Sending picture data stream should wait until this signal is de-asserted.
- FIFO_IN_WRITE_REQ_PIO
  - Asserted to write bitstream segment to FIFO in buffer. FIFO stores input data when this signal is asserted.
- FIFO_OUT_READ_REQ_PIO
  - Asserted when ARM wishes to read from the FIFO out. FIFO produces next data from the buffer when this signal is asserted.

Hardware Design contd.,

- Full source code for 8 bit and 24 bit FIFO will be provided
  - rle_fifo_8_24.v
- Use the RLE Verilog code that you implemented in Lab 3
- Top-level module
  - DE1_SOC_With_D5M.v
  - You will need make modifications to connect your new hardware
- Compile the design (Might take a while!)

Software design - preprocessing of the captured image

- Current image format uses 16 bits per pixel
- Black and white image only needs one bit to convey all information for the pixel
- Convert to a storage format that uses only 1 bit per pixel
- Ex:

Pixel 1: 0000 0000 0000 0001

Pixel 2: 0000 0000 0000 0000

Pixel 3: 0000 0000 0000 0001

…

Becomes

101…….

Software Design

- Use Altera HAL functions to send/receive data
  - alt_write_byte()
  - alt_read_byte()
  - alt_read_word()
- Send data to be compressed
  - while(buffer_not_full){
    alt_write_byte(data_pio_base, data);
    .....;
    }
- Receive data from compression
  - if(buffer_not_empty){
    data = alt_read_word(dataout_pio_base);
    ....;
    }

Software Design

- Software must be able to decompress data
  - Relatively easy
    - Data tells you what to write
    - Example: (10000000 00000000 00000111)
      - First Value = 1
      - Second Value = 7
      - Write 1, 7 times
      - Output: 1111111

Deliverables

- Implement a function in C to convert the captured color image to black and white on the ARM HPS and display the image on the monitor
- Compress this image using the RLE in FPGA fabric
- Implement a function in C to decompress the compressed image on the ARM HPS and display it on the monitor along with the compression ratio

Tips

- Go through the detailed instructions posted on the class website before starting the Lab 4

- Compressed data is variable length for every picture
  - Data could be compressed smaller, or expanded into larger data
  - Assume compressed data can be larger than the original picture data and do not fix the compressed output array size

Summary

- To implement hardware and software for RLE
  - Encoding is done by Hardware
    - GPIO is used
    - Read/Write data from/to hardware is done by using ALTERA API
  - Decoding is done by Software

- The RLE implementation is added to Lab 2
  - Take pictures and perform transformations
  - Display the images before and after compression and decompression occur

Extra credit

- Re-do lab 4 but with Linux installed on the DE1-SoC board
- Unlike bare-metal ARM HPS, Linux doesn’t have access to the physical memory space
- Use a Linux kernel function mmap which maps the physical memory space to the virtual memory space
- More resources will be posted in the future!

Questions

Appendix

How to Add/Wire New Modules

- Top Module wires the components together with the QSYS system
  - DE1_SOC_With_D5M.v is the top module for the project
  - Must add three components into the top module file
    - 2 Buffers
    - 1 RLE
- To add a module, must follow certain format
  - Module_File_Name Module_Name (
    .input1 (wire1)
    .input2 (wire 2)
    .output1 (wire3)
    );
- Computer_System is the QSYS system generated by QSYS
  - Look at QSYS – Generate – Show Instantiation Template to view names of connections to be added

Signals in Block Structure (Non -QSYS)

- FIFO_IN_READ_REQ
  - Asserted by the encoder when it is ready for the next bitstream segment.
- FIFO_IN_EMPTY
  - Indicates that the FIFO in buffer is empty. The RLE must wait until the FIFO in buffer is nonempty to request new data.
- FIFO_IN_ODATA[7:0]
  - Provides the next bit-stream segment to RLE unit when FIFO_IN_READ_REQ is asserted.
- RLE_OUT[23:0]
  - Passes the encoded data from the encoder to the FIFO out buffer.
- RLE_DONE
  - Indicates that the encoder unit is ready to output the next encoded segment. It is also used as a write assert signal for the FIFO out buffer.
- FIFO_OUT_FULL
  - Indicates that the FIFO out buffer is full. The RLE unit should wait until this signal is de-asserted to write its next prepared segment.